On Multi-route Maximum Flows in Networks

by
Charu C. Aggarwal
James B. Orlin

WP #3954-97 May 1997

On Multi-route Maximum Flows in Networks

by

Charu C. Aggarwal*
IBM T J Watson Research Center
Yorktown Heights, NY 10598
charu@watson.ibm.com

and

James B. Orlin
MIT Sloan School of Management
E53-357
Cambridge, MA 02139
jorlin@mit.edu

* Work done while at the MIT Operations Research Center

Abstract.

Let G = (N, A) be a network with a designated source node s, a designated sink node t, and a finite integral capacity $u_{ij} \leq U$ on each arc $(i,j) \in A$. An elementary K-flow is a flow of K units from s to t such that the flow on each arc is 0 or 1. A K-route flow is a flow from s to t that may be expressed as a non-negative linear sum of elementary K-flows. In this paper, we show how to determine a maximum K-route flow as a sequence of O(min{log KU, K}) minimum cut problems plus a single maximum flow problem. This improves upon the algorithm by Kishimoto, which solves this problem as a sequence of K minimum cut problems plus a maximum flow problem. In addition, we have simplified and extended some of the basic theory.

1. Introduction.

We study the multi-route flow problem, which models situations in which one needs to obtain a maximum flow from a source node s to a sink node t that is robust against physical failures on the arcs in the network. Let G = (N, A) be a network with n nodes including a source node s and a sink node t. Let m denote the number of arcs in A. Associated with each arc (i,j) is a capacity u_{ij}. We let $U = \max \{u_{ij} : (i,j) \in A\}$.

An *elementary K-flow* is defined to be a flow of one unit along K arc-disjoint paths from node s to node t. A *K-route flow* is a flow from node s to node t that can be expressed as a non-negative sum of elementary K-flows. In other words, a K-route flow may be decomposed into a set of elementary K-flows. The *value* of a K-route flow is equal to the total amount of flow entering the sink node t. The *maximum K-route flow problem* is the problem of determining a K-route flow with maximum flow value.

Because a K-route flow can be decomposed into elementary K-flows, the failure of any arc in the network will result in a reduction in the flow arriving at the sink by at most a factor of 1/K, without increasing the flow in any arc. To see this, observe that for each elementary K-flow in the decomposition, at most one of the K paths contains the failed arc. If one eliminates the flow on the path with the failed arc, the decrease in flow is by at most a factor of 1/K. Therefore, sending K-route flows provides a strong guarantee of protection against arc failures.

Kishimoto and Takeuchi [4] introduced the concept of 2-route flows, which was further studied by Kishimoto, Takeuchi, and Kishi [6]. Kishimoto and Takeuchi [5] and Kishimoto [3] considered the generalization to K-route flows. They provided both an elegant duality theory as well as efficient algorithms. As per Kishimoto [3], we define elementary flows that are arc disjoint s-t flows. Kishimoto observed that the theory

extends to elementary flows that are node disjoint s-t flows. His observation relies on a standard technique referred to as "node splitting."

In this paper, we simplify and extend the theory for K-route flows. Our theoretical development leads to two simple algorithms. The first algorithm solves the K-route flow problem as a sequence of log (KU) minimum s-t cut problems plus one additional maximum flow problem. The second algorithm solves the K-route flow problem as a sequence of at most K minimum s-t cut problems plus one additional maximum flow problem. Our algorithms improve upon Kishimoto's [3] algorithm, which has the same worst case running time as our second algorithm. Moreover, our second algorithm dominates Kishimoto's algorithm in the following sense: the number of minimum cut problems required as a subroutine by our second algorithm is at most the number of minimum cut problems required by Kishimoto's algorithm, and there are examples where the number is strictly less.

This paper is organized as follows. In the next section, we introduce some additional notation, and we review Kishimoto's algorithm for the K-route flow problem. In Section 3, we develop the theory underlying K-route flows. In Section 4, we present a generic algorithm for solving the maximum K-route flow problem as well as two implementations of the generic algorithm. In Section 5, we review Kishimoto and Takeuchi's [5] duality theory. A summary is presented in Section 6.

2. Kishimoto's Algorithm.

In this section, we introduce some additional notation and definitions, and then describe Kishimoto's algorithm.

An *s-t cut* is a partition of the node set N into two disjoint subsets S and $\bar{S}$ = N\S such that s $\in$ S and t $\in$ $\bar{S}$. The cut is denoted by (S, $\bar{S}$). The capacity of the cut (S, $\bar{S}$) is denoted as Cap(S), where

$$\text{Cap(S)} = \sum_{\substack{i \in S, j \notin S \\ (i,j) \in A}} u_{ij} .$$

For each value p and for each arc (i,j) we let $u_{ij}^{p} = \max\{p, u_{ij}\}$. We let G[p] denote the network G in which the arc capacities are replaced by u^p. We let Cut(p) denote the minimum value of a cut in network G[p]. And we let

$$\text{Cap(S,p)} = \sum_{\substack{i \in S, j \notin S \\ (i,j) \in A}} u_{ij}^{p} .$$

Thus, for each fixed p $\geq$ 0, Cut(p) = min{ Cap(S,p) : (S, $\bar{S}$) is an s-t cut }.

Kishimoto's Algorithm

begin

p_1 := U;

for i := 1 to K **do**

begin

if Cut(p_i) $\geq$ Kp_i **then** a maximum K-route flow is

a maximum s-t flow in G[p_i];

else p_{i+1} := (Cut(p_i) - (i-1) p_i)/ (K- i + 1);

end

end

When this procedure terminates with value p_i, then $Cut(p_i)$ is the maximum value of a K-route flow. Moreover, a maximum s-t flow x* in $G[p_i]$ is a maximum K-route flow. We refer the reader to Kishimoto [3] for a proof of the correctness of this algorithm.

The running time to find the maximum K-route flow is the time needed to solve K minimum s-t cut problems and one maximum flow problem. Kishimoto [3] expressed the running time as the time needed to solve K maximum flow problems. Here we observe that an s-t cut can be determined from a maximum s-t flow in O(m) additional steps, and so the asymptotic running time to determine a minimum capacity s-t cut is at most the time to determine a maximum s-t flow. The converse is not true. Given a minimum s-t cut, there is no algorithm that is guaranteed to determine a maximum s-t flow any faster than would be the case in starting from scratch. Moreover, a recent algorithm by Benczur and Karger [2] provides an approximate s-t cut in an undirected network in a time faster than any known approximation for the maximum flow problem. This algorithm suggests that the minimum s-t cut problem may be easier to solve than the maximum flow problem. Correspondingly, the algorithms presented in this paper solve the K-route flow problem as a sequence of minimum s-t cut problems plus a single maximum flow problem.

3. Characterizations of K-route Flows.

In this section, we characterize K-route flows. This characterization helps in establishing several fast algorithms for the K-route flow problem.

Lemma 1. If x' is a K-route flow with pK units arriving at the sink, then $x'_{ij} \leq p$ for all $(i,j) \in A$.

Proof. Suppose that x' is expressed as the linear sum of elementary K-route flows $y_1, \ldots, y_q$, with $x' = \Sigma_k \alpha_k y^k$. Then $\Sigma_k \alpha_k = p$, and hence

$$x'_{ij} = \Sigma_k \alpha_k y^p_{ij} \leq \Sigma_k \alpha_k \leq p \ . \qquad \blacklozenge$$

Theorem 1 below states that the converse of Lemma 1 is also true, that is, if x' is a flow with pK units arriving at the sink and if $x'_{ij} \leq p$ for all $(i,j) \in A$, then x' is also a K-route flow. Accordingly, Theorem 1 characterizes those flows that are K-route flows. We first establish a lemma that will be used in the proof of Theorem 1.

Lemma 2. Suppose that x is a feasible s-t flow for network G = (N,A) with v units of flow reaching the sink. Suppose that K is a positive integer and let $x_{max} = \max\{x_{ij} : (i,j) \in A\}$. If $x_{max} \leq v/K$, then there is an elementary K-route flow y such

i. if $x_{ij} = 0$, then $y_{ij} = 0$.
ii. if $x_{ij} = x_{max}$, then $y_{ij} = 1$.

Proof. Consider the maximum s-t flow problem P obtained by placing an upper bound on arc (i,j) by $\lceil x_{ij}/x_{max} \rceil$ and replacing a lower bound on arc (i,j) by $\lfloor x_{ij}/x_{max} \rfloor$. Then x/x_{max} is a feasible flow for problem P with flow value $v/x_{max} \geq K$ arriving at the sink node. Since all data are integral, there is a feasible integral solution for P as well with K units arriving at the sink. Any integral solution with a flow of at least K units arriving at the sink may be expressed via flow decomposition as the sum of flows on K paths from s to t plus the flows around directed cycles. These paths are disjoint because the maximum capacity of an arc is 1. Therefore, there is an elementary K-flow satisfying the properties of Lemma 2. $\blacklozenge$

For each flow x, let A[x] = {(i,j) : $x_{ij} > 0$}. That is, A[x] is the subset of arcs with positive flow. We say that x is *extreme* if A[x] has no directed cycles. In Theorem 1 below, we restrict attention to extreme flows. This restriction is without loss of generality since one may efficiently convert any flow x with flow value v into an extreme flow x' with flow value v' such that $x' \leq x$ and $v' \geq v$. (One simply cancels flow around cycles.)

Theorem 1. Suppose that x is a feasible extreme s-t flow with flow value v. Suppose that K is a positive integer. Then x is a K-route flow if and only if $x_{max} \leq v/K$.

Proof. The "only if" part was established in Lemma 1. So we need to prove the "if" part. If v = 0 then the only possible extreme flow is x = 0, and it follows that x is a K-route flow. (This is the only place in the proof where we explicitly use the fact that x is extreme.) So, we assume that v > 0.

We say that an arc (i,j) is intermediate if $0 < x_{ij} < x_{max}$. We will prove the result using induction on the number of intermediate arcs. We first establish the base of the induction by considering the case when there are no intermediate arcs. In this case, select a flow y as in Lemma 2. Then $y_{ij} = 1$ whenever $x_{ij} = x_{max}$, and $y_{ij} = 0$ whenever $x_{ij} = 0$. It follows that $x_{ij} = x_{max} y_{ij}$ for all (i,j), and the theorem is true.

The remaining case is the case in which v > 0, and the number of intermediate arcs is $k \geq 1$. We assume inductively that the theorem is true if the number of intermediate arcs is k-1 or fewer.

Let y denote the elementary K-flow as defined in Lemma 2.

Let $\Delta' = \max \{\Delta : 0 \leq x - \Delta y \leq x_{max} - \Delta\}$. Let $x' = x - \Delta' y$. Then $x'_{max} = x_{max} - \Delta'$,

and the flow from s to t in x' is v - KΔ'. Moreover, one can show that the number of intermediate arcs with respect to x' is strictly less than the number of intermediate arcs with respect to x. It follows inductively that x' is a K-route flow. Since x = x' + Δ'y, it follows that x is also a K-route flow. ◆

Theorem 1 gives a useful characterization of K-route flows. It states than an alternative characterization of a K-route flow is one such that the flow on any arc is at most 1/K of the total value of the flow from the source node s to the sink node t. Thus it is possible to develop a linear programming formulation for the K-route flow problem as follows:

Maximize v

subject to:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} v & \text{for } i = s \\ 0 & \text{for all } i \in N - \{s,t\} \\ -v & \text{for } i = t \end{cases}$$

$x_{ij} \le v/K$ for all $(i,j) \in A$

$0 \le x_{ij} \le u_{ij}$ for all $(i, j) \in A$.

Note that if the upper bounds "$x_{ij} \le v/K$" are removed from the formulation, then the resulting problem is a standard linear programming formulation of the maximum flow problem.

Theorem 1 characterizes the situations in which a flow is a K-route flow. We now characterize the optimum value of a K-route flow. Recall that Cut(p) is the smallest value of a cut in G[p].

Theorem 2. The maximum value of a K-route flow is the following:

$$z = \max\{Cut(p) : Cut(p) \geq Kp,\ 0 \leq p \leq U\}.$$

Proof. Let v denote the maximum value of a K-route flow. We first establish that $v \geq z$, and we subsequently establish that $v \leq z$.

Suppose that $z = Cut(p') = \max\{Cut(p) : Cut(p) \geq Kp,\ 0 \leq p \leq U\}$. Let x' denote the maximum flow from s to t in network G[p']. Let v' denote the amount of flow reaching the sink for flow x'. Then $v' = Cut(p') = z$, and by assumption $Cut(p') \geq Kp'$. Therefore, by Theorem 1, x' is a K-route flow. Then $z = v' \leq v$.

We now establish that $v \leq z$. Let x* be an optimal K-route flow, and let v denote the amount of the flow into node t. By Lemma 1, $x^*_{max} \leq v/K$. Therefore,

$$K x^*_{max} \leq v \leq Cut(x^*_{max}) \quad . \tag{1}$$

Finally, $Cut(x^*_{max}) \leq z = \max\{Cut(p) : Cut(p) \geq Kp\}$. We conclude that $v \leq z$, and so the theorem is proved. ◆

4. Algorithms for K-route Flows

Theorem 2 can be viewed as providing an algorithm for finding a maximum K-route flow.

Generic Algorithm.

begin

determine $p^* = \max\{p: \text{Cut}(p) \geq Kp, 0 \leq p \leq U\}$;

-- we will show how to determine p* in Algorithms 2 and 3 --

let x* be a maximum flow in G[p*];

end

Algorithm 1 is valid because of Theorem 1 and the fact that Cut(p) is monotone non-decreasing in p. The key unspecified detail for the algorithm is the determination of p*. We will provide two algorithms for determining p*. The first algorithm, which relies on binary search, is based on Lemmas 3 and 4 below.

Lemma 3. The function Cut(p) is monotone non-decreasing, piecewise linear, and concave.

Proof. The fact that $\text{Cut}(p') \geq \text{Cut}(p)$ for $p' \geq p$ is obvious. The fact that Cut( ) is piecewise linear and concave follows from the fact that it is the solution of a parametric linear program in minimization form. ◆

Lemma 4. Let $p^* = \max\{p : \text{Cut}(p) \geq Kp, 0 \leq p \leq U\}$. Then p* may be expressed as a rational number whose denominator is at most K.

Proof. If $p^* = U$, then p* is integral, and the result is true. So suppose that $p^* < U$. Choose an s-t cut $(S, \bar{S})$ and a very small positive number ε^* so that $(S, \bar{S})$ is a minimum cut in $G[p^*+\varepsilon]$ for all $\varepsilon \in [0,\varepsilon^*]$. The proof that the cut $(S, \bar{S})$ and the value ε^* exist follows from well known results in the theory of linear programming. Then

$$Cap(S,p^*) \geq Kp^* \tag{2a}$$

$$Cut(p^*+\varepsilon) = Cap(S, p^*+\varepsilon) < K(p^* + \varepsilon) \text{ for all } \varepsilon > 0. \tag{2b}$$

$$Cut(p^*+\varepsilon) - Cut(p^*) = Cap(S, p^* + \varepsilon) - Cap(S, p^*) = k^*\varepsilon \text{ for all } \varepsilon \in [0,\varepsilon^*], \tag{2c}$$

where k^* is the number of arcs directed from S to $\bar{S}$ with capacity strictly greater than p^*.

Inequalities (2a) and (2b) follow directly from the definition of p^*. Equation (2c) follows from the fact that $(S, \bar{S})$ is a minimum cut in $G[p^* + \varepsilon]$ for all $\varepsilon \in [0, \varepsilon^*]$.

From (2a) and (2b), and the continuity of Cap(S, p) in p, we conclude the following:

$$Cap(S, p^*) = Kp^*. \tag{2d}$$

From (2b), (2c), and (2d), we conclude that $0 \leq k^* < K$.

Finally, let $p' = \max \{u_{ij} : i \in S, j \in \bar{S}, (i,j) \in A, \text{ and } u_{ij} \leq p^*\}$. Thus p' is the greatest capacity of any arc in the cut whose capacity does not exceed p^*. Then by (2d) and the logic used to justify (2c),

$$Cap(S, p^*) = Cap(S, p') + k^*(p^*-p') = Kp^*. \tag{2e}$$

Solving for p^* in (2e), we get: $p^* = [Cap(S, p') + k^*p']/(K-k^*)$, which is a rational number whose denominator is at most K. Thus the theorem is true. ◆

We are now prepared to determine p^* using binary search.

Algorithm 1.

begin

use binary search to determine a lower bound L* and an upper bound U* such that

$L^* \leq p^* < U^*$, and $U^* - L^* < 1/K^2$;

choose a minimum cut $(S, \bar{S})$ for G[U*];

$p' = \max\{p : Cap(S, p) \geq Kp\}$;

-- then $p^* = p'$ --

find a maximum flow in G[p*];

end

Theorem 3. Algorithm 1 determines the maximum K-route flow by solving O(log KU) minimum cut problems and one maximum flow problem.

Proof. By Lemma 3, if Cut(p') < Kp', then Cut(p) < Kp' for all p > p'. So, one can search for the value p* using binary search. Initially, the size of the search interval is size U, and within $\log K^2U$ iterations, the size of the search interval is size at most K^{-2}. Therefore, the number of iterations is $O(\log K^2U) = O(\log KU)$.

We now establish that the algorithm terminates with an optimal K-route flow. Let $p' = \max\{p : Cap(S, p) \geq Kp\}$, where $(S, \bar{S})$ is the minimum cut for G[U*]. It suffices to show that $p' = p^* = \max\{p : Cut(p) \geq Kp\}$. Since $(S, \bar{S})$ is a minimum cut for G[U*], $Cut(U^*) = Cap(S, U^*) < KU^*$. So $p' < U^*$. By assumption, $Cut(p) \leq Cap(S, p) < Kp$ for $p > p'$. So $p' \geq p^*$ and $L^* \leq p^* \leq p' \leq U^*$. It follows that $| p' - p^* | < 1/K^2$. But p' and p* are both rational numbers with denominator at most K. (Recall that p' is a rational number with denominator at most K by Lemma 4.) When two distinct rational numbers a/b and a'/b' have denominators at most K, then their absolute difference is $| a'b - b'a | / bb' > K^{-2}$. So we conclude that p' and p* are not distinct; that is $p' = p^*$. ◆

The final algorithm of this paper is an algorithm that determines the value of p* by solving a sequence of O(K) minimum cut problems.

Algorithm 2.

begin

p' : = U;

choose a minimum cut $(S, \bar{S})$ for G[p'];

while Cap(S, p') < Kp' **do**

begin

p' := max{p: Cap(S, p) ≥ Kp};

let $(S, \bar{S})$ be a minimum cut for G[p'];

end

end

Theorem 4. Algorithm 2 requires the solution of O(K) min cut problems to determine p*.

Proof. Suppose that $(S, \bar{S})$ is the minimum cut for $G[p_1]$ as determined at some iteration of the while loop of Algorithm 2. Suppose that $p_2 = \max\{p : Cap(S, p) \geq Kp\}$, and let $(Q, \bar{Q})$ be the minimum cut for $G[p_2]$. Thus $(Q, \bar{Q})$ is the cut determined at the next time iteration of the while loop. If $Cap(Q, p_2) \geq Kp_2$, then the algorithm terminates with the cut $(Q, \bar{Q})$. We claim that this is the optimum solution to the K-route flow problem. To see this, recall that the maximum value is $z = \max\{Cut(p) : Cut(p) \geq Kp\}$. The algorithm selected p_2 so that $Cut(p) \leq Cap(S, p) < Kp$ for $p > p_2$. Moreover, by assumption, $Cut(p_2) = Kp_2$, and so $z = Cut(p_2)$ is the optimal value of a K-route flow. We now consider the case that $Cap(Q, p_2) < Kp_2$. We will show that this case can occur at most K times.

Let $g^-(S, p)$ denote the left derivative of Cap(S, p) evaluated at the point p. And let $g^+(S, p)$ be the right derivative of Cap(S, p) evaluated at the point p. We claim that $g^-(S, p_1) < g^+(Q, p_2) \leq g^-(Q, p_2)$ and that all are non-negative integers less than K. If these claims are correct, then the number of iterations of the while loop is at most K. We now establish that these claims are true.

Note that $g^-(Q, p)$ is the number of arcs in the cut set $(Q, \overline{Q})$ whose capacity is p or less. Since $Cap(Q, p_2) < Kp_2$, and g(Q, p) is concave in p, and g(Q, 0) = 0, it follows that $g^-(Q, p_1) < K$. Similarly, $g^+(Q, p)$ is the number of arcs in the cut set $(Q, \overline{Q})$ whose capacity is strictly greater than p. Therefore, $g^+(Q, p_2) \leq g^-(Q, p_2)$. So, it remains to show that $g^-(S, p_1) < g^+(Q, p_2)$.

We now claim that the following four inequalities are all valid:

$$Cut(p_1) \leq Cap(Q, p_1) \tag{3a}$$

$$Cap(Q, p_1) \leq Cap(Q, p_2) + g^+(Q, p_1)\,(p_1 - p_2) \tag{3b}$$

$$Cut(p_2) < Cap(S, p_2) \tag{3c}$$

$$Cap(S, p_2) \leq Cap(S, p_1) - g^-(S, p_1)\,(p_1 - p_2) \tag{3d}$$

Inequalities (3a) and (3c) follow from the assumptions. Inequalities (3b) and (3d) follow from the concavity of Cap(Q, p) and Cap(S, p). Adding inequalities (3a) - (3d), and substituting for $Cap(S, p_1)$ for $Cut(p_1)$ and substituting $Cap(Q, p_2)$ for $Cut(p_2)$ yields the inequality:

$$(g^{+}(Q, p_1) - g^{-}(S, p_1))(p_1 - p_2) > 0, \qquad (3e)$$

which completes the proof. ◆

5. Duality Theory.

In this section, we review the duality theory of Kishimoto and Takeuchi and provide a simple proof. We first recall that Theorem 2 characterizes the maximum value of a K-route flow as max {Cut(p) : Cut(p) ≥ Kp}. Also, Cut(p) = min{Cap(S, p) : $(S, \bar{S})$ is an s-t cut}.

For each cut $(S, \bar{S})$ and for each value $p \geq 0$, if Cap(S,p) < Kp, then Cut(p) < Kp, and so the maximum value of the K-route flow is strictly less than Cap(S,p). With this in mind, we define the *K-capacity* of the cut $(S, \bar{S})$ to be $\max_p\{Cap(S,p) : Cap(S,p) \geq Kp\}$.

Theorem 5. (Kishimoto and Takeuchi [5]). The maximum value of a K-route flow is equal to the minimum K-capacity of an s-t cut.

Proof. We have already seen that the maximum value of a K-route flow is at most the minimum K-capacity of an s-t cut. As before, let $p^* = \max \{p : Cut(p) \geq Kp\}$. Choose $(S, \bar{S})$ and $\varepsilon^* > 0$ so that $(S, \bar{S})$ is the minimum cut for $G[p^* + \varepsilon]$ for all $\varepsilon \in [0, \varepsilon^*]$. Then the K-capacity of the cut $(S, \bar{S})$ is $Cap(S, p^*) = Cut(p^*)$, and so we conclude that the K-capacity of the cut $(S, \bar{S})$ is the maximum value of a K-route flow. This completes the proof. ◆

6. Summary.

In this paper, we have characterized K-route flows and shown how to calculate the maximum K-route flow as a sequence of O(min {log KU, K}) minimum cut problems plus a single maximum flow problem. This improves upon earlier results by Kishimoto.

Acknowledgments.

We gratefully acknowledge partial support through a grant from the UPS Foundation and from ONR grant ONR N00014-96-1-0051. We also thank Ravi Ahuja for his suggestions leading to improvements in the exposition.

References.

[1] R.K. Ahuja, T. L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, N.J. 1993.

[2] A. Benczur and D. Karger. Approximate s-t min cuts in $O(n^2)$ time, STOC 1996, 47-55.

[3] W. Kishimoto. "A method for obtaining the maximum multi-route flow in a network. To appear in *Networks*.

[4] W. Kishimoto and M. Takeuchi. "On two-route flows in an undirected network. IEICE Technical Report, CAS90-19, DSP-90-23 (1990).

[5] W. Kishimoto and M. Takeuchi. "On m-route flows in an undirected network. *IEICE Transactions* **J-76-A**, 1993, 1185-1200 .

[6] W. Kishimoto and M. Takeuchi, and G. Kishi. "Two-route flows in an undirected flow network. *IEICE Transactions* **J-75-A**, 1992, 1699-1717 (in Japanese).